Correlating Formal Semantic Models of Reo Connectors: Connector Coloring and Constraint Automata

Sung-Shik T.Q. Jongmans
Centrum Wiskunde & Informatica (CWI)
Amsterdam, the Netherlands
jongmans@cwi.nl

Farhad Arbab
Centrum Wiskunde & Informatica (CWI)
Amsterdam, the Netherlands
farhad.arbab@cwi.nl

Over the past decades, coordination languages have emerged for the specification and implementation of interaction protocols for communicating software components. This class of languages includes Reo, a platform for compositional construction of connectors. In recent years, various formalisms for describing the behavior of Reo connectors have come to existence, each of them serving its own purpose. Naturally, questions about how these models relate to each other arise. From a theoretical point of view, answers to these questions provide us with better insight into the fundamentals of Reo, while from a more practical perspective, these answers broaden the applicability of Reo's development tools. In this paper, we address one of these questions: we investigate the equivalence between coloring models and constraint automata, the two most dominant and practically relevant semantic models of Reo. More specifically, we define operators that transform one model to the other (and vice versa), prove their correctness, and show that they distribute over composition. To ensure that the transformation operators map one-to-one (instead of many-to-one), we extend coloring models with data constraints. Though primarily a theoretical contribution, we sketch some potential applications of our results: the broadening of the applicability of existing tools for connector verification and animation.

1 Introduction

Over the past decades, *coordination languages* have emerged for the specification and implementation of interaction protocols for communicating software components. This class of languages includes Reo [1], a platform for compositional construction of *connectors*. Connectors in Reo (or *circuits*) form the communication mediums through which components can interact with each other. Essentially, Reo circuits impose constraints on the order in which components can send and receive *data items* to and from each other. Although ostensibly simple, Reo connectors can describe complex protocols (e.g., a solution to the Dining Philosophers problem [2]). In recent years, various formal models for describing the behavior of Reo circuits have arisen, including a coalgebraic model [4], various operational models (e.g., *constraint automata* [6]), and two coloring models [8] (we mention more models in Section 7). Each of these formalisms serves its own purpose: the coalgebraic model has become Reo's reference semantics, constraint automata play a dominant role in connector verification (e.g., the Vereofy model checker [5]), and the coloring models facilitate the animation of connectors (e.g., the implementation of the work in Chapter 6 of [9] in the Eclipse Coordination Tools).

Having this wide variety of semantic models, questions about how they relate to each other naturally arise. We identify two reasons for why this question, moreover, requires answering. First, from a purely theoretical point of view, answers provide us with better and possibly new insights into Reo's fundamentals. Second, from a more practical perspective, such answers broaden the applicability of tools—both existing and future—that assist developers in designing their Reo circuits. For instance, the

Bliudze, S., Bruni, R., Carbone, M., Silva, A. (Eds.); ICE 2011
EPTCS 59, 2011, pp. 84–103, doi:10.4204/EPTCS.59.8

correspondence between constraint automata and the coloring model with two colors (the topic of this paper) enables us to, on the one hand, model check connectors with a coloring model as their formal semantics, and, on the other, animate connectors with a constraint automaton as their behavioral model. We consider this important because contemporary tools for verification and animation cannot operate on, respectively, coloring models and constraint automata.

Contributions We investigate the relation between coloring models with two colors and constraint automata. We show how to transform the former to the latter and demonstrate bi-similarity between an original and its transformation. In the opposite direction, we show how to transform constraint automata to equivalent coloring models, prove that these transformations define each other's inverse, and again show bi-similarity. Additionally, we prove the compositionality of our transformation operators. To ensure that our transformation operators map one-to-one (instead of many-to-one), we extend coloring models with data constraints. To illustrate the practical relevance of our work, we sketch one of its applications: the integration of verification and animation of context-sensitive connectors in Vereofy. We emphasize, however, that with this paper, we aim at establishing equivalences: we consider it primarily a theoretical contribution and a formal foundation for future tool development.

Organization In Section 2, we discuss preliminaries of Reo. In Section 3, we extend coloring models with data constraints. In Section 4, we present a transformation from such data-aware coloring models to constraint automata, and in Section 5, we present a transformation in the opposite direction. In Section 6, we sketch an application of our results. Section 7 concludes the paper and includes related work.

2 Connector Structures, Coloring Models, and Constraint Automata

In this section, we discuss the essentials of Reo (relevant to this paper): the structure of circuits and two formal models of its behavior. Henceforth, we write "connector" or "circuit" to refer to both the structure and the intended behavior of a communication medium between software components.

2.1 Connector Structures

We start with the structure of circuits. A Reo connector consists of *nodes* through which data items can *flow*. We distinguish three types of nodes: *input nodes* on which components can issue *write requests* for data items, *internal nodes* that the connector uses to internally route data items, and *output nodes* on which components can issue *take requests* for data items. We call input and output nodes collectively, the *boundary nodes* of a connector. Write and take requests, collectively called *I/O requests*, remain *pending* on a boundary node until they *succeed*, in which case their respective nodes *fire*. We describe the structure of a connector formally as a set of nodes, typically denoted as N, and a binary relation on these nodes, typically denoted as E. We use this relation to specify the direction of the flow through the nodes in N: if $\langle n_{in}, n_{out} \rangle \in E$, this means that node n_{in} can route incoming data items from itself to n_{out}.

Definition 1 (Universe of nodes)**.** Node *is the set of nodes.*

Definition 2 (Connector structure)**.** *A connector structure* σ *is a pair* $\langle N, E \rangle$ *with* $N \subseteq \mathsf{Node}$ *a set of nodes and* $E \subseteq N \times N$ *a relation such that* $n \in N$ *implies* $\langle n_{in}, n \rangle \in E$ *or* $\langle n, n_{out} \rangle \in E$.

The side condition in the previous definition ensures that a connector structure does not include superfluous nodes through which data items never flow (note that it does allow for cyclic structures). We

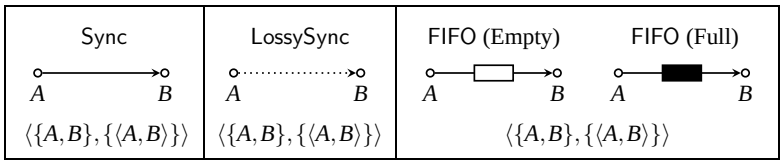

Figure 1: Pictorial representation and formal definition of the structure of Sync, LossySync, and FIFO.

associate the following sets and definitions with a connector structure $\sigma = \langle N, E \rangle$. First, we define the sets of its input, output, and internal nodes.

$$\begin{array}{lcl} \texttt{input}_\sigma & = & \{\, n \in N \mid \langle n_{in}, n \rangle \notin E \,\} \\ \texttt{output}_\sigma & = & \{\, n \in N \mid \langle n, n_{out} \rangle \notin E \,\} \\ \texttt{internal}_\sigma & = & \{\, n \in N \mid \langle n_{in}, n \rangle, \langle n, n_{out} \rangle \in E \,\} \end{array}$$

Note that these three definitions specify mutually disjoint sets and that their union equals N (due to the side condition in Definition 2). If $\texttt{internal}_\sigma = \emptyset$, the circuit whose topology σ describes belongs to the class of connectors called *primitives*, the most elementary mediums between components.

To illustrate the previous definition, Figure 1 shows pictorial representations and formal definitions of the structures of three common (binary) primitives. Because the pictorial representations of these primitives may give away some hints about their behavior, we discuss these informally here; the formal definitions appear later in this section. The Sync primitive consists of an input node and an output node. Data items flow through this primitive only if both of its nodes have pending I/O requests. The LossySync primitive behaves similarly, but loses a data item if its input node has a pending write request while its output node has no pending take request. In contrast to the previous two primitives, connectors can have *buffers* to store data items in. Such connectors exhibit different states, while the internal configuration of Sync and LossySync always stays the same. For instance, the FIFO primitive consists of an input node, an output node, and a buffer of size 1. In its EMPTY state, a write request on the input node of FIFO causes a data item to flow into its buffer—i.e., this buffer becomes full—while a take request on its output node remains pending. Conversely, in its FULL state, a write request on its input node remains pending, while a take request on its output node causes a data item to flow from the buffer to this output node—i.e., the buffer becomes empty. Finally, note the equality of the formal definitions of the structures of Sync, LossySync, and FIFO: $\langle \{A,B\}, \{\langle A,B \rangle\} \rangle$. In general, all primitives that route data items from a single input node to a single output node have this structure (up to node renaming).

We can construct complex connectors from simpler constituents (e.g., instances of primitives) using *composition*. Connector structures σ_1 and σ_2 can compose if each of their shared nodes serves as an input node in σ_1 and as an output node in σ_2 or vice versa. To compose such compatible connector structures, we merge their sets of nodes and E relations.

Definition 3 (Composition of connector structures)**.** *Let $\sigma_1 = \langle N_1, E_1 \rangle$ and $\sigma_2 = \langle N_2, E_2 \rangle$ be connector structures such that $N_1 \cap N_2 = (\texttt{input}_{\sigma_1} \cap \texttt{output}_{\sigma_2}) \cup (\texttt{input}_{\sigma_2} \cap \texttt{output}_{\sigma_1})$. Their composition, denoted $\sigma_1 \boxtimes \sigma_2$, is a connector structure defined as:*

$$\sigma_1 \boxtimes \sigma_2 = \langle N_1 \cup N_2, E_1 \cup E_2 \rangle$$

To illustrate the previous definition, Figure 2 shows the pictorial representation and formal definition of the structure of LossyFIFO, a connector composed of LossySync and FIFO. The LossyFIFO connector consists of one input node, one internal node, and one output node. Similar to FIFO, the LossyFIFO connector exhibits the states EMPTY and FULL. Informally, in the EMPTY state, a write request on the input node of LossyFIFO *always* causes a data item to flow into its buffer, while a take request on its

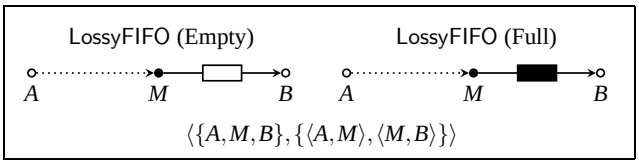

Figure 2: Pictorial representation and formal definition of the structure of LossyFIFO.

output node remains pending. In the FULL state, a write request on its input node *always* causes a data item to flow from its input node towards its buffer, but gets lost before reaching its internal node; a take request on its output node causes a data item to flow from its buffer to this output node.

Later in this paper, we use the possible compatibility between two connector structures as a well-formedness condition to define the composition of various *comprehensive* models—models that combine a connector structure with a behavioral model—of Reo connectors.

2.2 Coloring Models

We proceed with *coloring models* [8, 9] as the first semantic model we discuss. Coloring models work by marking nodes of a connector with *colors* that specify whether data items flow through these nodes or not. Depending on the number of colors, different models with different levels of expressiveness arise. In this paper, we assume a total of two colors: the *flow color* ——— (data items can flow through the nodes it marks) and the *no-flow color* - - - - (data items cannot flow through the nodes it marks).[1] To describe a single behavior alternative of a connector in a given state, we define *colorings*: maps from sets of nodes to sets of colors, which assign to each node in the set a color that indicates whether this node fires (or not) in the behavior that the coloring describes. We collect all behavior alternatives of a connector in sets of colorings called *coloring tables*.

Definition 4 (Colors [8])**.** $\mathsf{Color} = \{$ ———, - - - - $\}$ *is a set of colors.*

Definition 5 (Coloring [8])**.** *Let* $N \subseteq \mathsf{Node}$*. A coloring* c *over* N *is a map* $N \to \mathsf{Color}$*.*

Definition 6 (Coloring table [8])**.** *Let* $N \subseteq \mathsf{Node}$*. A coloring table* T *over* N *is a set* $\{N \to \mathsf{Color}\}$ *of colorings over* N*.*

To accommodate connectors that exhibit different behavior in different states (e.g., connectors with buffers to store data items in), we use *coloring table maps* (CTM): maps from sets of *indexes* (representing the states of a connector) to sets of coloring tables (describing the allowed behavior alternatives in these states).[2] Subsequently, to model the change of state a connector incurs when (some of) its nodes fire, we use *next functions*. The next function of a connector maps an index λ in the domain Λ of a CTM S and a coloring in the coloring table to which S maps λ to some index in Λ (possibly the same λ).

Definition 7 (Universe of indexes)**.** Index *is the set of indexes.*

Definition 8 (Coloring table map [9])**.** *Let* $N \subseteq \mathsf{Node}$ *and* $\Lambda \subseteq \mathsf{Index}$*. A coloring table map* S *over* $[N,\Lambda]$ *is a total map* $S : \Lambda \to 2^{\{N \to \mathsf{Color}\}}$ *from indexes to coloring tables over* N*.*

Definition 9 (Next function [9])**.** *Let* S *be a CTM over* $[N,\Lambda]$*. A next function* η *over* S *is a partial map* $\Lambda \times \{N \to \mathsf{Color}\} \rightharpoonup \Lambda$ *from [index, coloring]-pairs to indexes such that* $[\lambda, c \mapsto \lambda'] \in \eta$ *iff* $c \in S(\lambda)$*.*

[1] Two colors yield the *2-coloring model*. Alternatively, the *3-coloring model* features three different colors to mark nodes with. Although many believe that the 3-coloring model has a higher degree of expressiveness than the 2-coloring model, a recent investigation [10] suggests otherwise.

[2] In [9], Costa calls coloring table maps *indexed sets of coloring tables*.

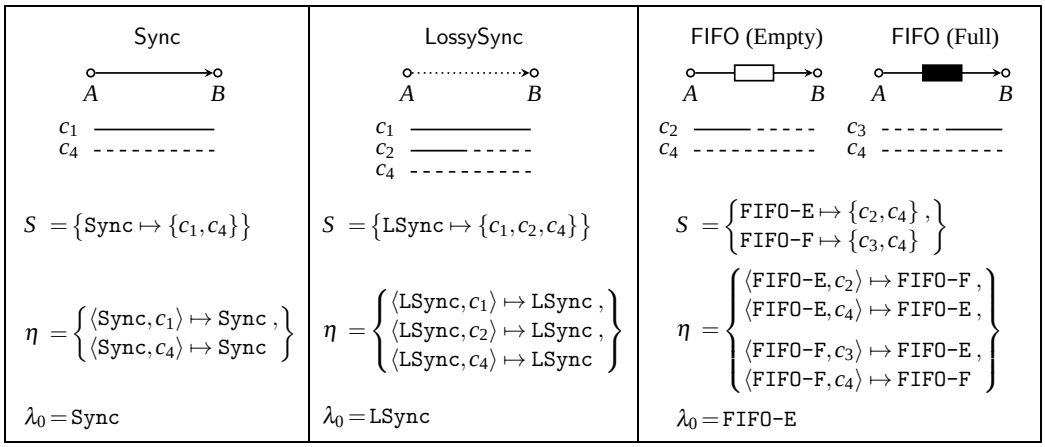

Figure 3: Colorings, CTMs, and initialized next functions of Sync, LossySync, and FIFO.

Next, we slightly extend the connector coloring framework as presented in [8, 9] by introducing *initialized next functions*. Suppose a CTM S over $[N, \Lambda]$ and a next function η over S: an *initialization* of η—formally a pair of a next function and an index—associates η with some $\lambda_0 \in \Lambda$. This λ_0 represents the initial state of the connector whose behavior η models.

Definition 10 (Initialized next function)**.** *Let S be a CTM over $[N, \Lambda]$. An initialized next function ε over S is a pair $\langle \eta, \lambda_0 \rangle$ with η a next function over S and $\lambda_0 \in \Lambda$.*

Finally, we join coloring models—i.e., initialized next functions, which comprehensively describe the behavior of circuits—and connector structures in *ε-connectors*: complete formal models of connectors.

Definition 11 (ε-connector)**.** *Let $N \subseteq \mathsf{Node}$ and let S be a CTM over $[N, \Lambda]$. An ε-connector $\mathscr{C}^{\mathsf{Col}}$ over $[N, S]$ is a pair $\langle \sigma, \varepsilon \rangle$ with $\sigma = \langle N, E \rangle$ a connector structure and ε an initialized next function over S.*

To illustrate the previous definitions, Figure 3 shows the coloring models that describe the behavior of Sync, LossySync, and FIFO, whose structures we depicted in Figure 1.[3]

When we compose two connectors that have coloring models as formal semantics, we can compute the coloring model of the composed connector by composing the coloring models of its constituents. We describe this composition process in a bottom-up fashion. First, to compose two *compatible colorings*—i.e., colorings that assign the same colors to their shared nodes—we merge the domains of these colorings and map each node *n* in the resulting set to the color that one of the colorings assigns to *n*. The composition of two coloring tables then comprises the computation of a new coloring table that contains the pairwise compositions of the compatible colorings in the two individual coloring tables.

Definition 12 (Composition of colorings [8])**.** *Let c_1 and c_2 be colorings over N_1 and N_2 such that $c_1(n) = c_2(n)$ for all $n \in N_1 \cap N_2$. Their composition, denoted by $c_1 \cup c_2$, is a coloring over $N_1 \cup N_2$ defined as:*

$$c_1 \cup c_2 = \left\{ n \mapsto \kappa \,\middle|\, n \in N_1 \cup N_2 \text{ and } \kappa = \begin{pmatrix} c_1(n) & \text{if } n \in N_1 \\ c_2(n) & \text{otherwise} \end{pmatrix} \right\}$$

[3] Because a composed connector may consist of multiple instances of the same (primitive) connector, we should actually use indexes that enable distinguishing between such instances. For example, rather than just "Sync", we could extend this index with a distinctive subscript, such as the set of nodes that the respective Sync primitive connects (e.g., "$\texttt{Sync}_{A,B}$") or, alternatively, an integer (e.g., "$\texttt{Sync}_1$"). Throughout this paper, however, we abstract from such subscripts for notational convenience.

Definition 13 (Composition of coloring tables [8])**.** *Let T_1 and T_2 be coloring tables over N_1 and N_2. Their composition, denoted by $T_1 \cdot T_2$, is a coloring table over $N_1 \cup N_2$ defined as:*

$$T_1 \cdot T_2 = \{\, c_1 \cup c_2 \mid c_1 \in T_1 \text{ and } c_2 \in T_2 \text{ and } c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2 \,\}$$

Next, the composition of two CTMs comprises the computation of a new CTM that maps each pair of indexes in the Cartesian product of the domains of the two individual CTMs to the composition of the coloring tables to which these CTMs map the indexes in the pair. We define the composition of two next functions in terms of the Cartesian product, the composition of colorings, and the composition of CTMs.

Definition 14 (Composition of CTMs [9])**.** *Let S_1 and S_2 be CTMs over $[N_1, \Lambda_1]$ and $[N_2, \Lambda_2]$. Their composition, denoted by $S_1 \odot S_2$, is a CTM over $[N_1 \cup N_2, \Lambda_1 \times \Lambda_2]$ defined as:*

$$S_1 \odot S_2 = \{\, \langle \lambda_1, \lambda_2 \rangle \mapsto S_1(\lambda_1) \cdot S_2(\lambda_2) \mid \lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2 \,\}$$

Definition 15 (Composition of next functions [9])**.** *Let η_1 and η_2 be next functions over S_1 and S_2 with S_1 and S_2 defined over $[N_1, \Lambda_1]$ and $[N_2, \Lambda_2]$. Their composition, denoted by $\eta_1 \otimes \eta_2$, is a next function over $S_1 \odot S_2$ defined as:*

$$\eta_1 \otimes \eta_2 = \left\{ \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle, c_1 \cup c_2 \\ \downarrow \\ \langle \eta_1(\lambda_1, c_1), \eta_2(\lambda_2, c_2) \rangle \end{array} \middle| \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and} \\ c_1 \cup c_2 \in (S_1 \odot S_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\}$$

We define the composition of initialized next functions in terms of the composition of next functions and take the pair of the initial states as the initial state of the composition.

Definition 16 (Composition of initialized next functions)**.** *Let $\varepsilon_1 = \langle \eta_1, \lambda_0^1 \rangle$ and $\varepsilon_2 = \langle \eta_2, \lambda_0^2 \rangle$ be initialized next functions over S_1 and S_2. Their composition, denoted by $\varepsilon_1 \otimes \varepsilon_2$, is an initialized next function over $S_1 \odot S_2$ defined as:*

$$\varepsilon_1 \otimes \varepsilon_2 = \langle \eta_1 \otimes \eta_2, \langle \lambda_0^1, \lambda_0^2 \rangle \rangle$$

Finally, we define the composition of ε-connectors in terms of the composition of connector structures and the composition of initialized next functions.

Definition 17 (Composition of ε-connectors)**.** *Let $\mathscr{C}_1^{\mathsf{Col}} = \langle \sigma_1, \varepsilon_1 \rangle$ and $\mathscr{C}_2^{\mathsf{Col}} = \langle \sigma_2, \varepsilon_2 \rangle$ be ε-connectors over $[N_1, S_1]$ and $[N_2, S_2]$ such that $\sigma_1 \boxtimes \sigma_2$ is defined. Their composition, denoted by $\mathscr{C}_1^{\mathsf{Col}} \times \mathscr{C}_2^{\mathsf{Col}}$, is an ε-connector over $[N_1 \cup N_2, S_1 \odot S_2]$ defined as:*

$$\mathscr{C}_1^{\mathsf{Col}} \times \mathscr{C}_2^{\mathsf{Col}} = \langle \sigma_1 \boxtimes \sigma_2, \varepsilon_1 \otimes \varepsilon_2 \rangle$$

To illustrate the previous definitions, Figure 4 shows a coloring model that describes the behavior of LossyFIFO, whose structure we depicted in Figure 2. In this figure, the index of a coloring specifies its origin: a coloring c_{ij} results from composing c_i (of LossySync) with c_j (of FIFO) in Figure 3.[4]

2.3 Constraint Automata

We end this section with *constraint automata* (CA) [6] as the second semantic model we discuss. A CA consists of a (possibly singleton) set of states, which correspond one-to-one to the states of the connector whose behavior it models and a set of transitions between them; in contrast to standard automata, CA

[4] Note that this composed coloring model of LossyFIFO does *not* describe its intended behavior as outlined near the end of the previous subsection: coloring c_{24}, allowed in the EMPTY state according to the composed model, describes the loss of a data item between *A* and *M*. We, however, consider this inadmissible behavior in the EMPTY state. Clarke et al. recognize this inconsistency between intuition and formal practice in [8] and call it the problem of describing *context-sensitive* connectors. We refer the reader interested in the challenges that context-sensitive connectors entail to [7, 8, 9, 10].

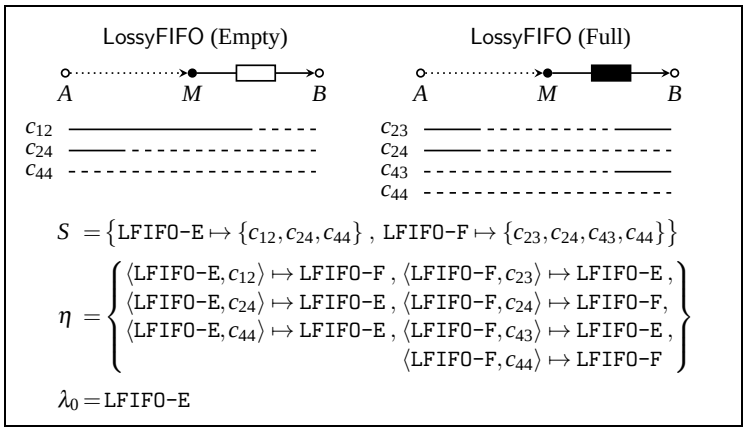

Figure 4: Colorings, CTM, and next function of LossyFIFO. We abbreviate $\langle \texttt{LSync}, \texttt{FIFO-E}\rangle$ by $\texttt{LFIFO-E}$ and $\langle \texttt{LSync}, \texttt{FIFO-F}\rangle$ by $\texttt{LFIFO-F}$.

do not have accepting states. A transition of a CA carries a label that consists of two elements: a set of nodes and a *data constraint*. The former, called a *firing set*, describes which nodes fire simultaneously in the state the transition leaves from; the latter specifies the conditions that the content of the data items that flow through these firing nodes must satisfy. To define (the universe of) data constraints, we assume a universe of data items. This set contains every data item that we may send through a Reo connector.

Definition 18 (Universe of data items)**.** Data *is the set of data items.*

Definition 19 (Universe of data constraints [6])**.** Constraint *is the set of data constraints such that each* $g \in$ Constraint *complies with the following grammar:*

$$g ::= g \wedge g \mid \neg g \mid \top \mid \#n = d \text{ with } n \in \mathsf{Node} \text{ and } d \in \mathsf{Data}$$

Informally, $\#n$ means "the data item that flows through n", while $\wedge$, $\neg$, and $\top$ have their usual meaning. For convenience, we also allow their derived Boolean operators such as $\vee$, $\Rightarrow$ (implication), etc., as syntactic sugar; we adopt $\#n_1 = \#n_2$ (with $n_1, n_2 \in \mathsf{Node}$) as an abbreviation of $\bigvee_{d \in \mathsf{Data}}(\#n_1 = d \wedge \#n_2 = d)$. This gives us sufficient machinery to define CA. Recall that CA serve as operational models of connector behavior—their states correspond one-to-one to the states of a connector, while their transitions specify for each state when and what data items can flow through which nodes.

Definition 20 (Constraint automaton [6])**.** *Let* $N \subseteq$ Node *and* $G \subseteq$ Constraint*. A constraint automaton* α *over* $[N,G]$ *is a tuple* $\langle Q, R, q_0\rangle$ *with* Q *a set of states,* $R \subseteq Q \times 2^N \times G \times Q$ *a transition relation, and* $q_0 \in Q$ *an initial state.*

As initialized next functions, CA comprehensively model circuit behavior. Similar to ε-connectors, therefore, we introduce α*-connectors*: pairs that consist of a connector structure and a CA.

Definition 21 (α-connector)**.** *Let* $N \subseteq$ Node *and* $G \subseteq$ Constraint*. An* α*-connector* $\mathscr{C}^{\mathsf{CA}}$ *over* $[N,G]$ *is a pair* $\langle \sigma, \alpha\rangle$ *with* $\sigma = \langle N, E\rangle$ *a connector structure and* $\alpha = \langle Q, R, q_0\rangle$ *a CA over* $[N,G]$.

To illustrate the previous definitions, the CA of Sync, LossySync, and FIFO appear in Figure 5. For simplicity, we assume the universe of data items a singleton with the string "foo" as its sole element. In general, the CA of FIFO contains a distinct state for each data item in Data that may occupy its buffer. Note that rather than naming states symbolically (e.g., $q, p, q_0, q_1, \ldots$), we name states by the same indexes we encountered previously when defining coloring table maps in coloring models. Henceforth,

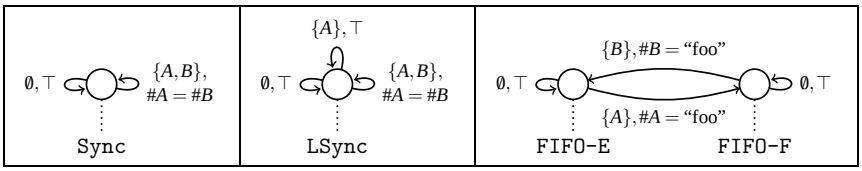

Figure 5: Constraint automata of Sync, LossySync, and FIFO with Data = {"foo"}.

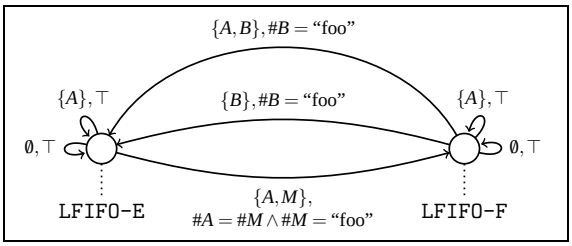

Figure 6: Constraint automaton of LossyFIFO with Data = {"foo"}.

without loss of generality, we assume that if $\alpha = \langle Q, R, q_0 \rangle$ denotes a CA, $Q \subseteq \mathsf{Index}$ (Footnote 3 still applies).

When we compose two connectors that have CA as their behavioral model, we can compute the CA of the composed connector by composing the CA of its constituents: the binary operator for CA composition takes the Cartesian product of the set of states of its arguments, designates the pair of their initial states as the initial state of the composed CA, and computes a new transition relation.

Definition 22 (Composition of CA [6])**.** *Let $\alpha_1 = \langle Q_1, R_1, q_0^1 \rangle$ and $\alpha_2 = \langle Q_2, R_2, q_0^2 \rangle$ be CA over $[N_1, G_1]$ and $[N_2, G_2]$. Their composition, denoted $\alpha_1 \bowtie \alpha_2$, is a CA over $[N_1 \cup N_2, G_1 \wedge G_2]$[5] defined as:*

$$\alpha_1 \bowtie \alpha_2 = \langle Q_1 \times Q_2, R, \langle q_0^1, q_0^2 \rangle \rangle$$

$$\textit{with: } R = \left\{ \langle \langle q_1, q_2 \rangle, F_1 \cup F_2, g_1 \wedge g_2, \langle q_1', q_2' \rangle \rangle \;\middle|\; \begin{array}{l} \langle q_1, F_1, g_1, q_1' \rangle \in R_1 \textit{ and } \langle q_2, F_2, g_2, q_2' \rangle \in R_2 \\ \textit{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\}$$

The previous definition differs slightly from the one in [6]: we do not *implicitly* assume that all states have a silent τ-transition (as in [6]), but *explicitly* include these transitions in our models (represented by transitions labeled with $\langle \emptyset, \top \rangle$). Though essentially a matter of representation, it simplifies later proofs.

Definition 23 (Composition of α-connectors)**.** *Let $\mathscr{C}_1^{\mathsf{CA}} = \langle \sigma_1, \alpha_1 \rangle$ and $\mathscr{C}_2^{\mathsf{CA}} = \langle \sigma_2, \alpha_2 \rangle$ be CA connectors over $[N_1, G_1]$ and $[N_2, G_2]$ such that $\sigma_1 \boxtimes \sigma_2$ is defined. Their composition, denoted $\mathscr{C}_1^{\mathsf{CA}} \times \mathscr{C}_2^{\mathsf{CA}}$, is an α-connector over $[N_1 \cup N_2, G_1 \wedge G_2]$ defined as:*

$$\mathscr{C}_1^{\mathsf{CA}} \times \mathscr{C}_2^{\mathsf{CA}} = \langle \sigma_1 \boxtimes \sigma_2, \alpha_1 \bowtie \alpha_2 \rangle$$

To illustrate the previous definition, we depict the CA of LossyFIFO in Figure 6.[6]

In the remainder, with a slight loss of generality, we consider only deterministic CA. We believe, however, that this limits the applicability of our results only marginally: in practice, non-deterministic CA occur rarely. In fact, even after almost a decade of research and literature on Reo, we have not encountered a (primitive) connector whose behavior one cannot describe concisely with a deterministic CA.[7] As a final remark, we emphasize that our presentation of CA remains superficial: we covered only the essentials relevant to the rest of this paper. A more comprehensive overview appears in [6].

[5] For notational convenience, we write $G_1 \wedge G_2$ for $\{ g_1 \wedge g_2 \mid g_1 \in G_1 \text{ and } g_2 \in G_2 \}$.

[6] Similar to the coloring model of LossyFIFO in Figure 4, its CA does not model its intended semantics: the transition $\langle \texttt{LFIFO-E}, \{A\}, \top, \texttt{LFIFO-E} \rangle$ describes the inadmissible loss of data in the EMPTY state (similar to c_{24} in Figure 4).

[7] Many non-deterministic connectors, in contrast, *do* exist, but we can model their behavior with deterministic CA.

3 Data-Aware Coloring Models

In this section, we make traditional coloring models *data-aware* by extending them with constraints similar to those carried by transitions of constraint automata. We introduce this extension, because one of the transformation operators that we define later in this paper lacks a desirable property otherwise: it would map many-to-one instead of one-to-one. More precisely, the transformation from α-connectors to ε-connectors would map *different* α-connectors—i.e., those whose transitions carry different data constraints but equal firing sets—to the *same* ε-connector. Alternatively, to gain this desirable one-to-one property, we could have narrowed the scope of this paper to a special class of CA whose members abstract from data constraints: the transitions of these *port automata* (PA) [11] carry only a firing set. We favor the extension of coloring models for generality (note that CA subsume PA).

We extend coloring models with data-awareness by associating each coloring with a data constraint from $\mathsf{Constraint}$ (recall from Definitions 19 and 20 that data constraints in CA come from the same universe). Such a *constraint coloring* describes a computation step of a connector wherein (i) data flows through the nodes marked by the flow color and (ii) the data constraint holds. Below we give the formal definition. With respect to notation, we write the symbols that denote constituents of data-aware coloring models in ***font*** (but use the same letters as for coloring models without constraints).

Definition 24 (Constraint coloring)**.** *Let $N \subseteq \mathsf{Node}$ and $G \subseteq \mathsf{Constraint}$. A constraint coloring $\boldsymbol{c}$ over $[N,G]$ is a pair $\langle c,g\rangle$ with c a coloring over N and $g \in G$ a data constraint.*

Note that our definition does not exclude a constraint coloring $\boldsymbol{c} = \langle c,g\rangle$ over $[N,G]$ with inconsistent c and g. For example, c may mark some node $n \in N$ with the no-flow color, while $g \in G$ entails the flow of a data item $d \in \mathsf{Data}$ through n. We do not forbid such constraint colorings, because they do not impair the models in which they appear: they merely describe behavior that cannot arise in practice.

Next, we incorporate data constraints in the definitions of the other constituents of ordinary coloring models (as presented in Section 2.2). This turns out straightforwardly. To summarize the upcoming definitions: (i) a *constraint coloring table* is a set of constraint colorings, (ii) a *constraint CTM* is a map from indexes to constraint coloring tables, (iii) a *constraint next function* is map from [index, constraint coloring]-pairs to indexes, (iv) an *initialized constraint next function* is a [constraint next function, index]-pair, and (v) an ***$\boldsymbol{\varepsilon}$-connector*** is a formal model of a connector that has an initialized constraint next function as its behavioral model.

Definition 25 (Constraint coloring table)**.** *Let $N \subseteq \mathsf{Node}$ and $G \subseteq \mathsf{Constraint}$. A constraint coloring table $\boldsymbol{T}$ over $[N,G]$ is a (sub)set (of) $\{N \to \mathsf{Color}\} \times G$ of constraint colorings over $[N,G]$.*

Definition 26 (Constraint CTM)**.** *Let $N \subseteq \mathsf{Node}$, $G \subseteq \mathsf{Constraint}$, and $\Lambda \subseteq \mathsf{Index}$. A constraint CTM $\boldsymbol{S}$ over $[N,G,\Lambda]$ is a map $\Lambda \to 2^{\{N\to\mathsf{Color}\}\times G}$ from indexes to constraint coloring tables over $[N,G]$.*

Definition 27 (Constraint next function)**.** *Let $\boldsymbol{S}$ be a constraint CTM over $[N,G,\Lambda]$. A constraint next function $\boldsymbol{\eta}$ over $\boldsymbol{S}$ is a partial map $\Lambda \times (\{N \to \mathsf{Color}\} \times G) \rightharpoonup \Lambda$ from [index, constraint coloring]-pairs to indexes such that $[\langle \lambda, \boldsymbol{c}\rangle \mapsto \lambda'] \in \boldsymbol{\eta}$ iff $\boldsymbol{c} \in \boldsymbol{S}(\lambda)$.*

Definition 28 (Initialized constraint next function)**.** *Let $\boldsymbol{S}$ be a constraint CTM over $[N,G,\Lambda]$. An initialized constraint next function $\boldsymbol{\varepsilon}$ over $\boldsymbol{S}$ is a pair $\langle \boldsymbol{\eta}, \lambda_0\rangle$ with $\boldsymbol{\eta}$ a constraint next function over $\boldsymbol{S}$ and $\lambda_0 \in \Lambda$.*

Definition 29 (***$\boldsymbol{\varepsilon}$***-connector)**.** *Let $\boldsymbol{S}$ be a constraint CTM over $[N,G,\Lambda]$. An $\boldsymbol{\varepsilon}$-connector $\mathscr{C}^{\mathsf{Col}}$ over $[N,\boldsymbol{S}]$ is a pair $\langle \sigma, \boldsymbol{\varepsilon}\rangle$ with $\sigma = \langle N,E\rangle$ a connector structure and $\boldsymbol{\varepsilon}$ an initialized constraint next function over $\boldsymbol{S}$.*

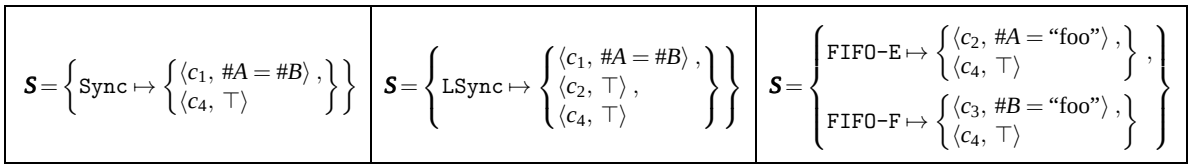

Figure 7: Constraint CTMs of Sync, LossySync, and FIFO with $\mathsf{Data} = \{\text{“foo”}\}$.

To illustrate the previous definitions, Figure 7 shows the constraint CTMs of Sync, LossySync, and FIFO. The colorings c_i with $i \in \{1,2,3,4\}$ refer to the colorings in Figure 3. For instance, constraint coloring $\langle c_1, \#A = \#B \rangle$ in the constraint CTM of Sync (respectively LossySync) describes the behavior alternative of Sync (respectively LossySync) wherein the same data item flows through A and B. Constraint coloring $\langle c_4, \top \rangle$, present in all constraint CTMs, describes the behavior alternative wherein a connector idles. Due to the constraint $\top$, this may always happen. Similarly, LossySync can always behave as described by $\langle c_2, \top \rangle$, but whereas $\langle c_4, \top \rangle$ entails no flow at all, $\langle c_2, \top \rangle$ entails flow through A and no flow through B. Here, $\top$ specifies that we do not care about which data item flows through A. With respect to FIFO, for simplicity of the example, we assume the universe of data items a singleton similar to Section 2.3. In general, the constraint CTM of FIFO contains a distinct constraint coloring table for each data item in Data that may occupy the buffer.

Finally, we must update the composition operators for coloring models to incorporate data constraints. For brevity, we give these definitions only for constraint colorings and constraint coloring tables. The composition operators for constraint CTMs (symbol: $\odot$), (initialized) constraint next functions (symbol: $\otimes$), and $\boldsymbol{\varepsilon}$-connectors (symbol: $\times$) resemble their respective composition operators in Section 2.2: essentially, it suffices to replace S_1, S_2, η_1, η_2, ε_1, and ε_2 in Definitions 14–17 with their ***font*** versions $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_2$, $\boldsymbol{\varepsilon}_1$, and $\boldsymbol{\varepsilon}_2$. We require only these minor updates, because our extension of coloring models with data constraints affects only the definition of colorings directly. For completeness, in Appendix A, we give the definitions of those composition operators that we skip below.

Definition 30 (Composition of constraint colorings)**.** *Let $\boldsymbol{c}_1 = \langle c_1, g_1 \rangle$ and $\boldsymbol{c}_2 = \langle c_2, g_2 \rangle$ be colorings over $[N_1, G_1]$ and $[N_2, G_2]$ such that $c_1(n) = c_2(n)$ for all $n \in N_1 \cap N_2$. Their composition, denoted by $\boldsymbol{c}_1 \cup \boldsymbol{c}_2$, is a constraint coloring over $[N_1 \cup N_2, G_1 \wedge G_2]$[5] defined as:*

$$\boldsymbol{c}_1 \cup \boldsymbol{c}_2 = \langle c_1 \cup c_2, g_1 \wedge g_2 \rangle$$

Definition 31 (Composition of constraint coloring tables)**.** *Let $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ be constraint coloring tables over $[N_1, G_1]$ and $[N_2, G_2]$. Their composition, denoted by $\boldsymbol{T}_1 \cdot \boldsymbol{T}_2$, is a constraint coloring table over $[N_1 \cup N_2, G_1 \wedge G_2]$ defined as:*

$$\boldsymbol{T}_1 \cdot \boldsymbol{T}_2 = \{\, \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \mid \boldsymbol{c}_1 = \langle c_1, g_1 \rangle \in \boldsymbol{T}_1 \text{ and } \boldsymbol{c}_2 = \langle c_2, g_2 \rangle \in \boldsymbol{T}_2 \text{ and } c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2 \,\}$$

Note that the composition operator for CA in Definition 22 computes the label of a composite transition by taking the conjunction of two data constraints, similar to how we handle the combination of data constraints in Definition 30. The lemmas that we formulate and prove in the subsequent sections establish the appropriateness of taking the conjunction of data constraints in the context of coloring models. To illustrate the previous definitions, Figure 8 shows the constraint CTM of LossyFIFO. The colorings c_i with $i \in \{12, 24, 43, 44\}$ refer to the colorings in Figure 4.

Recently, in [14], Proença uses pairs of colorings and *node-to-data functions* as transition labels of his *behavioral automata* to account for the transfer of data that takes place through data-flows described by those colorings. This suggests using [coloring, node-to-data function]-pairs as a data-aware coloring model. However, our constraint-based extension offers a more concise formalization. For instance,

$$\boldsymbol{S} = \left\{ \texttt{LFIFO-E} \mapsto \left\{ \begin{array}{l} \langle c_{12},\ \#A = \#M \wedge \#M = \text{“foo”} \rangle\ , \\ \langle c_{24},\ \top \rangle\ , \\ \langle c_{44},\ \top \rangle \end{array} \right\}\ ,\ \texttt{LFIFO-F} \mapsto \left\{ \begin{array}{l} \langle c_{12},\ \#B = \text{“foo”} \rangle\ , \\ \langle c_{24},\ \top \rangle\ , \\ \langle c_{43},\ \#B = \text{“foo”} \rangle\ , \\ \langle c_{44},\ \top \rangle \end{array} \right\} \right\}$$

Figure 8: Constraint CTM of LossyFIFO with Data = {“foo”}. We abbreviate $\langle \texttt{LSync}, \texttt{FIFO-E} \rangle$ by LFIFO-E and $\langle \texttt{LSync}, \texttt{FIFO-F} \rangle$ by LFIFO-F.

in Proença's model, to give the semantics of a Sync primitive, one must include a separate [coloring, node-to-data function]-pair in its coloring table for each data item in Data. With our constraint-based extension, in contrast, we capture this with a single constraint coloring as shown in Figure 7.

4 From $\boldsymbol{\varepsilon}$-Connectors to α-Connectors

In this section, we present a unary operator, denoted by $\mathbb{L}$, which takes as argument an $\boldsymbol{\varepsilon}$-connector and produces an *equivalent* α-connector; shortly, we elaborate on the meaning of “equivalance” in this context. We call our process of transforming an $\boldsymbol{\varepsilon}$-connector to an α-connector the $\mathbb{L}$-*transformation*. By defining the $\mathbb{L}$-transformation for *any* $\boldsymbol{\varepsilon}$-connector, it follows that the class of connectors that we can model as α-connector *includes* those that we can model as $\boldsymbol{\varepsilon}$-connector—i.e., constraint automata are *at least* as expressive as data-aware coloring models.

The $\mathbb{L}$-operator works as follows; suppose we wish to transform an $\boldsymbol{\varepsilon}$-connector $\mathscr{C}^{\textsf{Col}} = \langle \sigma, \boldsymbol{\varepsilon} \rangle$ over $[N, \boldsymbol{S}]$ with $\boldsymbol{S}$ a constraint CTM over $[N, G, \Lambda]$. Whereas the connector structure σ does not incur any change (because $\mathbb{L}$ alters only the behavioral model), from the initialized constraint next function $\boldsymbol{\varepsilon} = \langle \boldsymbol{\eta}, \lambda_0 \rangle$, the $\mathbb{L}$-operator derives a constraint automaton. First, $\mathbb{L}$ instantiates the set of states of this derived CA with the set of indexes Λ: this seems reasonable as Λ denotes the set of indexes that represent the states of the connector that $\mathscr{C}^{\textsf{Col}}$ models. Next, $\mathbb{L}$ constructs a transition relation R based on the mappings in $\boldsymbol{\eta}$: for each $[\langle \lambda, \langle c, g \rangle \rangle \mapsto \lambda'] \in \boldsymbol{\eta}$, the $\mathbb{L}$-operator creates a transition from state λ to state λ', labeled with g as its data constraint and with the set of nodes to which c assigns the flow color as its firing set. Finally, λ_0 becomes the initial state of the new CA.

Definition 32 ($\mathbb{L}$ for $\boldsymbol{\varepsilon}$-connectors)**.** *Let $\mathscr{C}^{\textsf{Col}} = \langle \sigma, \boldsymbol{\varepsilon} \rangle$ be an $\boldsymbol{\varepsilon}$-connector over $[N, \boldsymbol{S}]$ with $\boldsymbol{\varepsilon} = \langle \boldsymbol{\eta}, \lambda_0 \rangle$ and $\boldsymbol{S}$ a constraint CTM over $[N, G, \Lambda]$. The $\mathbb{L}$-transformation of $\mathscr{C}^{\textsf{Col}}$, denoted by $\mathbb{L}(\mathscr{C}^{\textsf{Col}})$, is defined as:*

$$\mathbb{L}(\mathscr{C}^{\textsf{Col}}) = \langle \sigma, \mathbb{L}(\boldsymbol{\varepsilon}) \rangle$$

with: $\mathbb{L}(\boldsymbol{\varepsilon}) = \langle \Lambda, R, \lambda_0 \rangle$

and: $R = \{\ \langle \lambda, F, g, \boldsymbol{\eta}(\lambda, \boldsymbol{c}) \rangle \mid \lambda \in \Lambda \text{ and } \boldsymbol{c} = \langle c, g \rangle \in \boldsymbol{S}(\lambda) \text{ and } F = \{\ n \in N \mid c(n) = \text{——}\ \}\ \}$

The following proposition states that the application of $\mathbb{L}$ to an η-connector yields an α-connector.

Proposition 1. *Let $\mathscr{C}^{\textsf{Col}}$ be an $\boldsymbol{\varepsilon}$-connector over $[N, \boldsymbol{S}]$ with $\boldsymbol{S}$ defined over $[N, G, \Lambda]$. Then, $\mathbb{L}(\mathscr{C}^{\textsf{Col}})$ is an α-connector over $[N, G]$.*

Proof. Let $\mathscr{C}^{\textsf{Col}} = \langle \sigma, \boldsymbol{\varepsilon} \rangle$ with $\boldsymbol{\varepsilon} = \langle \boldsymbol{\eta}, \lambda_0 \rangle$. Then, by Definition 32, $\mathbb{L}(\mathscr{C}^{\textsf{Col}}) = \langle \sigma, \mathbb{L}(\boldsymbol{\varepsilon}) \rangle = \langle \sigma, \langle \Lambda, R, \lambda_0 \rangle \rangle$. By Definition 21, we must show that $\langle \Lambda, R, \lambda_0 \rangle$ is a CA over $[N, G]$. To demonstrate this, by Definition 20, we must show that $R \subseteq \Lambda \times 2^N \times G \times \Lambda$. Because, by the premise, $\mathscr{C}^{\textsf{Col}}$ is an $\boldsymbol{\varepsilon}$-connector over $[N, \boldsymbol{S}]$, by Definition 29, $\boldsymbol{\varepsilon}$ is an initialized constraint next function over $\boldsymbol{S}$, hence, by Definition 28, the co-domain of $\boldsymbol{\eta}$ is Λ. Also, by Definition 32, for all $\langle \lambda, F, g, \boldsymbol{\eta}(\lambda, \boldsymbol{c}) \rangle \in R$ with $\boldsymbol{c} = \langle c, g \rangle$, it holds that $\lambda \in \Lambda$, $F \subseteq N$, and $g \in G$ (this latter follows from Definition 24). □

In the rest of this section, we prove the equivalence between an $\boldsymbol{\varepsilon}$-connector $\mathscr{C}^{\textsf{Col}}$ and the α-connector that results from applying $\mathbb{L}$ to $\mathscr{C}^{\textsf{Col}}$. Additionally, we prove the distributivity of $\mathbb{L}$ over composition.

4.1 Correctness of $\mathbb{L}$

In this subsection, we prove the *correctness* of $\mathbb{L}$: we consider $\mathbb{L}$ correct if its application to an $\boldsymbol{\varepsilon}$-connector yields an *equivalent* α-connector. We call an $\boldsymbol{\varepsilon}$-connector and an α-connector equivalent if there exists a *bi-simulation relation* that relates these two connector models. Informally, an α-connector $\mathscr{C}^{\mathsf{CA}}$ is bi-similar to an $\boldsymbol{\varepsilon}$-connector $\mathscr{C}^{\mathsf{Col}}$ if, for each mapping in the constraint next function of $\mathscr{C}^{\mathsf{Col}}$, there exists a *corresponding* transition in the CA of $\mathscr{C}^{\mathsf{CA}}$—i.e., a transition that describes the same behavior in terms of the nodes that fire, the data items that flow, and the change of state—and vice versa.

Definition 33 (Bi-simulation)**.** *Let $\mathscr{C}^{\mathsf{CA}} = \langle \sigma, \alpha \rangle$ with $\alpha = \langle Q, R, q_0 \rangle$ be an α-connector over $[N, G]$ and $\mathscr{C}^{\mathsf{Col}} = \langle \sigma, \boldsymbol{\varepsilon} \rangle$ with $\boldsymbol{\varepsilon} = \langle \boldsymbol{\eta}, \lambda_0 \rangle$ an $\boldsymbol{\varepsilon}$-connector over $[N, \boldsymbol{S}]$ with $\boldsymbol{S}$ defined over $[N, G, \Lambda]$. $\mathscr{C}^{\mathsf{CA}}$ and $\mathscr{C}^{\mathsf{Col}}$ are bi-similar, denoted as $\mathscr{C}^{\mathsf{CA}} \sim \mathscr{C}^{\mathsf{Col}}$, if there exists a relation $\mathscr{R} \subseteq Q \times \Lambda$ such that $\langle q_0, \lambda_0 \rangle \in \mathscr{R}$ and for all $\langle q, \lambda \rangle \in \mathscr{R}$:*

(I) *If $\langle q, F, g, q' \rangle \in R$ then there exists a $\lambda' \in \Lambda$ such that:*

- $[\langle \lambda, \boldsymbol{c} \rangle \mapsto \lambda'] \in \boldsymbol{\eta}$ *with* $\boldsymbol{c} = \langle c, g \rangle$*;*
- $\langle q', \lambda' \rangle \in \mathscr{R}$*;*
- $F = \{\, n \in N \mid c(n) = \text{——} \,\}$.

(II) *If $[\langle \lambda, \boldsymbol{c} \rangle \mapsto \lambda'] \in \boldsymbol{\eta}$ with $\boldsymbol{c} = \langle c, g \rangle$ then there exists a $q' \in Q$ such that:*

- $\langle q, F, g, q' \rangle \in R$*;*
- $\langle q', \lambda' \rangle \in \mathscr{R}$*;*
- $F = \{\, n \in N \mid c(n) = \text{——} \,\}$.

In that case, $\mathscr{R}$ is called a bi-simulation relation.

Next, we formulate and prove Lemma 1, which states the bi-similarity between an $\boldsymbol{\varepsilon}$-connector $\mathscr{C}^{\mathsf{Col}}$ and its $\mathbb{L}$-transformation $\mathbb{L}(\mathscr{C}^{\mathsf{Col}})$. In our proof, we choose the diagonal relation on the set of indexes as a candidate bi-simulation relation.

Lemma 1. *Let $\mathscr{C}^{\mathsf{Col}}$ be an $\boldsymbol{\varepsilon}$-connector. Then, $\mathbb{L}(\mathscr{C}^{\mathsf{Col}}) \sim \mathscr{C}^{\mathsf{Col}}$.*

Proof. Let $\mathscr{C}^{\mathsf{Col}} = \langle \sigma, \boldsymbol{\varepsilon} \rangle$ with $\boldsymbol{\varepsilon} = \langle \boldsymbol{\eta}, \lambda_0 \rangle$ be defined over $[N, \boldsymbol{S}]$ with $\boldsymbol{S}$ defined over $[N, G, \Lambda]$. Additionally, let $\mathbb{L}(\mathscr{C}^{\mathsf{Col}}) = \langle \sigma, \mathbb{L}(\boldsymbol{\varepsilon}) \rangle$ with $\mathbb{L}(\boldsymbol{\varepsilon}) = \langle \Lambda, R, \lambda_0 \rangle$. We show that $\mathscr{R} = \{\, \langle \lambda, \lambda \rangle \mid \lambda \in \Lambda \,\}$ is a bi-simulation relation by demonstrating that it satisfies (I) and (II) of Definition 33. Let $\langle \lambda, \lambda \rangle \in \mathscr{R}$.

(I) Suppose $\langle \lambda, F, g, \lambda' \rangle \in R$. Then, by Definition 32 of $\mathbb{L}$, there exists a $\boldsymbol{c} = \langle c, g \rangle \in \boldsymbol{S}(\lambda)$ such that $\lambda' = \boldsymbol{\eta}(\lambda, \boldsymbol{c})$. Hence, $[\langle \lambda, \boldsymbol{c} \rangle \mapsto \lambda'] \in \boldsymbol{\eta}$. Also, by the definition of $\mathscr{R}$, $\langle \lambda', \lambda' \rangle \in \mathscr{R}$. Finally, by Definition 32 of $\mathbb{L}$, $F = \{\, n \in N \mid c(n) = \text{——} \,\}$. Therefore, $\mathscr{R}$ satisfies (I).

(II) Suppose $[\langle \lambda, \boldsymbol{c} \rangle \mapsto \lambda'] \in \boldsymbol{\eta}$ with $\boldsymbol{c} = \langle c, g \rangle$. Then, by Definition 27 of $\boldsymbol{\eta}$, it holds that $\lambda \in \Lambda$ and $\boldsymbol{c} \in \boldsymbol{S}(\lambda)$, hence by Definition 32 of $\mathbb{L}$, $\langle \lambda, F, g, \lambda' \rangle \in R$ with $F = \{\, n \in N \mid c(n) = \text{——} \,\}$. Also, by the definition of $\mathscr{R}$, $\langle \lambda', \lambda' \rangle \in \mathscr{R}$. Therefore, $\mathscr{R}$ satisfies (II).

Thus, $\mathscr{R}$ satisfies (I) and (II). Also, because $\lambda_0 \in \Lambda$ by Definition 28, $\langle \lambda_0, \lambda_0 \rangle \in \mathscr{R}$. Hence, $\mathscr{R}$ is a bi-simulation relation. Therefore, $\mathbb{L}(\mathscr{C}^{\mathsf{Col}}) \sim \mathscr{C}^{\mathsf{Col}}$. □

4.2 Distributivity of $\mathbb{L}$

We end this section with the distributivity (compositionality) lemma of $\mathbb{L}$. Informally, it states that it does not matter whether we first compose $\boldsymbol{\varepsilon}$-connectors $\mathscr{C}_1^{\mathsf{Col}}$ and $\mathscr{C}_2^{\mathsf{Col}}$ and then apply $\mathbb{L}$ to the composition or first apply $\mathbb{L}$ to $\mathscr{C}_1^{\mathsf{Col}}$ and $\mathscr{C}_2^{\mathsf{Col}}$ and then compose the transformations; the resulting α-connectors equal each other. The relevance of this result lies in the potential reduction in the amount of overhead that it allows for when applying the $\mathbb{L}$-operator in practice. This works as follows. There exist tools for Reo that operate on constraint automata and that have built-in functionality for their composition. By the distributivity lemma of $\mathbb{L}$, to use such a tool, we need to transform Reo's common primitives only once, store these in a library, and use this library together with the built-in functionality for composition to

construct the CA of composed connectors (on which the tool subsequently operates). Thus, the overhead of this approach remains constant. In contrast, the overhead of the alternative—i.e., first composing coloring models and then transforming the resulting composition using $\mathbb{L}$—grows linear in the number of composed connectors one wishes to apply the tool on.[8] In Section 6, we illustrate the foregoing with a concrete example; here, we proceed with the lemma and our proof, which, although rather technical, essentially consists of a series of straightforward applications of definitions that allow us to rewrite the transition relations of the composed automata.

Lemma 2. *Let $\mathscr{C}_1^{\mathsf{Col}}$ and $\mathscr{C}_2^{\mathsf{Col}}$ be $\boldsymbol{\varepsilon}$-connectors. Then, $\mathbb{L}(\mathscr{C}_1^{\mathsf{Col}}) \times \mathbb{L}(\mathscr{C}_2^{\mathsf{Col}}) = \mathbb{L}(\mathscr{C}_1^{\mathsf{Col}} \times \mathscr{C}_2^{\mathsf{Col}})$.*

Proof. Let $\mathscr{C}_1^{\mathsf{Col}} = \langle \sigma_1, \boldsymbol{\varepsilon}_1 \rangle$ and $\mathscr{C}_2^{\mathsf{Col}} = \langle \sigma_2, \boldsymbol{\varepsilon}_2 \rangle$ with $\boldsymbol{\varepsilon}_1 = \langle \boldsymbol{\eta}_1, \lambda_0^1 \rangle$ and $\boldsymbol{\varepsilon}_2 = \langle \boldsymbol{\eta}_2, \lambda_0^2 \rangle$ be defined over $[N_1, \boldsymbol{S}_1]$ and $[N_2, \boldsymbol{S}_2]$. Applying Definition 32 of $\mathbb{L}$, Definition 23 of $\times$, and Definition 39 of $\boldsymbol{\times}$ (informally on page 93) yields:

$$\langle \sigma_1 \boxtimes \sigma_2, \mathbb{L}(\boldsymbol{\varepsilon}_1) \bowtie \mathbb{L}(\boldsymbol{\varepsilon}_2) \rangle = \langle \sigma_1 \boxtimes \sigma_2, \mathbb{L}(\boldsymbol{\varepsilon}_1 \otimes \boldsymbol{\varepsilon}_2) \rangle$$

We focus on proving $\mathbb{L}(\boldsymbol{\varepsilon}_1) \bowtie \mathbb{L}(\boldsymbol{\varepsilon}_2) = \mathbb{L}(\boldsymbol{\varepsilon}_1 \otimes \boldsymbol{\varepsilon}_2)$. Applying Definition 32 of $\mathbb{L}$ to the left-hand side (LHS) and Definition 38 of $\otimes$ (informally on page 93) to the right-hand side (RHS) yields:

$$\begin{array}{c} \langle \Lambda_1, R_1, \lambda_0^1 \rangle \\ \bowtie \\ \langle \Lambda_2, R_2, \lambda_0^2 \rangle \end{array} = \mathbb{L} \left(\left\langle \left\{ \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle, \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \\ \downarrow \\ \langle \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1), \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \rangle \end{array} \middle| \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \text{and} \\ \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\}, \langle \lambda_0^1, \lambda_0^2 \rangle \right\rangle \right)$$

$$\text{with: } R_1 = \left\{ \langle \lambda_1, F_1, g_1, \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \rangle \middle| \begin{array}{l} \lambda_1 \in \Lambda_1 \text{ and } \boldsymbol{c}_1 = \langle c_1, g_1 \rangle \in \boldsymbol{S}_1(\lambda_1) \text{ and} \\ F_1 = \{ n \in N_1 \mid c_1(n) = \text{——} \} \end{array} \right\}$$

$$\text{and: } R_2 = \left\{ \langle \lambda_2, F_2, g_2, \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \rangle \middle| \begin{array}{l} \lambda_2 \in \Lambda_2 \text{ and } \boldsymbol{c}_2 = \langle c_2, g_2 \rangle \in \boldsymbol{S}_1(\lambda_2) \text{ and} \\ F_2 = \{ n \in N_2 \mid c_2(n) = \text{——} \} \end{array} \right\}$$

Applying Definition 22 of $\bowtie$ to the LHS, and Definition 32 of $\mathbb{L}$ to the RHS yields:

$$\langle \Lambda_1 \times \Lambda_2, R, \langle \lambda_0^1, \lambda_0^2 \rangle \rangle = \langle \Lambda_1 \times \Lambda_2, R', \langle \lambda_0^1, \lambda_0^2 \rangle \rangle$$

$$\text{with: } R = \left\{ \langle \langle \lambda_1, \lambda_2 \rangle, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1', \lambda_2' \rangle \rangle \middle| \begin{array}{l} \langle \lambda_1, F_1, g_1, \lambda_1' \rangle \in R_1 \text{ and } \langle \lambda_2, F_2, g_2, \lambda_2' \rangle \in R_2 \\ \text{and } F_1 \cap N_2 = F_2 \cap N_1 \end{array} \right\}$$

$$\text{and: } R' = \left\{ \langle \lambda, F, g, (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)(\lambda, \boldsymbol{c}) \rangle \middle| \begin{array}{l} \lambda \in \Lambda_1 \times \Lambda_2 \text{ and } \boldsymbol{c} = \langle c, g \rangle \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\lambda) \\ \text{and } F = \{ n \in N_1 \cup N_2 \mid c(n) = \text{——} \} \end{array} \right\}$$

What remains to be shown is $R = R'$. This follows from Figure 9. □

Although we consider only coloring models with two colors in this paper, one can apply Definition 32 of $\mathbb{L}$ also to 3-colored $\boldsymbol{\varepsilon}$-connectors. In fact, Lemma 1 (bi-simulation) would still hold! Essentially, this means that 3-colored $\boldsymbol{\varepsilon}$-connectors do not have a higher degree of expressiveness than coloring models with two colors. In contrast, Lemma 2 (compositionality) does *not* hold if we consider 3-colored $\boldsymbol{\varepsilon}$-connectors. More precisely, the fourth—counted from top to bottom—equality in Figure 9 ("Because, by the definition of F_1 and F_2...") becomes invalid if we consider coloring models with three colors. This means that, although coloring models with two and three colors have the same degree of expressiveness, *they compose differently*: paradoxically, the addition of a third color restricts, as intended, the number of compatible colorings. This allows us, for instance, to describe compositional context-sensitive connectors with three colors (considered impossible with two colors).

[8] In the current exposition, we assume composing two $\boldsymbol{\varepsilon}$-connectors and two α-connectors have equal costs. In Section 6, we argue for the merits of our approach when the cost of coloring model composition differs from the cost of CA composition.

$$
\begin{array}{l}
\quad R \\
= \text{/* By the definition of } R \text{ in Lemma 2 */} \\
\quad \{\, \langle\langle \lambda_1,\lambda_2\rangle, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \mid \langle \lambda_1, F_1, g_1, \lambda_1'\rangle \in R_1 \text{ and } \langle \lambda_2, F_2, g_2, \lambda_2'\rangle \in R_2 \text{ and } F_1 \cap N_2 = F_2 \cap N_1 \,\} \\
= \text{/* By the definitions of } R_1 \text{ and } R_2 \text{ in Lemma 2, and by introducing } \lambda = \langle \lambda_1,\lambda_2\rangle \text{ */} \\
\quad \left\{ \langle \lambda, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \lambda_1 \in \Lambda_1 \text{ and } \boldsymbol{c}_1 = \langle c_1, g_1\rangle \in \boldsymbol{S}_1(\lambda_1) \text{ and } \lambda_1' = \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \text{ and } F_1 = \{\, n \in N_1 \mid c_1(n) = \text{——} \,\} \text{ and} \\ \lambda_2 \in \Lambda_2 \text{ and } \boldsymbol{c}_2 = \langle c_2, g_2\rangle \in \boldsymbol{S}_2(\lambda_2) \text{ and } \lambda_2' = \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \text{ and } F_2 = \{\, n \in N_2 \mid c_2(n) = \text{——} \,\} \text{ and} \\ F_1 \cap N_2 = F_2 \cap N_1 \text{ and } \lambda = \langle \lambda_1, \lambda_2\rangle \end{array} \right\} \\
= \text{/* Because, by the Cartesian product, } [\lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2] \text{ iff } \langle \lambda_1,\lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \text{ */} \\
\quad \left\{ \langle \lambda, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \boldsymbol{c}_1 = \langle c_1, g_1\rangle \in \boldsymbol{S}_1(\lambda_1) \text{ and } \lambda_1' = \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \text{ and } F_1 = \{\, n \in N_1 \mid c_1(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_2 = \langle c_2, g_2\rangle \in \boldsymbol{S}_2(\lambda_2) \text{ and } \lambda_2' = \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \text{ and } F_2 = \{\, n \in N_2 \mid c_2(n) = \text{——} \,\} \text{ and} \\ F_1 \cap N_2 = F_2 \cap N_1 \text{ and } \lambda = \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \end{array} \right\} \\
= \text{/* Because, by the definition of } F_1 \text{ and } F_2 \text{ in Lemma 2, } [F_1 \cap N_2 = F_2 \cap N_1] \text{ iff } [\{\, n \in N_1 \cap N_2 \mid c_1(n) = \text{——} \,\} = \{\, n \in N_1 \cap N_2 \mid c_2(n) = \text{——} \,\}], \text{ and because, as } c_1 \text{ and } c_2 \text{ are 2-colorings, } [\{\, n \in N_1 \cap N_2 \mid c_1(n) = \text{——} \,\} = \{\, n \in N_1 \cap N_2 \mid c_2(n) = \text{——} \,\}] \text{ iff } [c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2] \text{ */} \\
\quad \left\{ \langle \lambda, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \boldsymbol{c}_1 = \langle c_1, g_1\rangle \in \boldsymbol{S}_1(\lambda_1) \text{ and } \lambda_1' = \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \text{ and } F_1 = \{\, n \in N_1 \mid c_1(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_2 = \langle c_2, g_2\rangle \in \boldsymbol{S}_2(\lambda_2) \text{ and } \lambda_2' = \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \text{ and } F_2 = \{\, n \in N_2 \mid c_2(n) = \text{——} \,\} \text{ and} \\ [c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2] \text{ and } \lambda = \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \end{array} \right\} \\
= \text{/* Because, by Definition 31 of } \cdot, \; [\boldsymbol{c}_1 = \langle c_1, g_1\rangle \in \boldsymbol{S}_1(\lambda_1) \text{ and } \boldsymbol{c}_2 = \langle c_2, g_2\rangle \in \boldsymbol{S}_2(\lambda_2) \text{ and } c_1(n) = c_2(n) \text{ for all } n \in N_1 \cap N_2] \text{ iff } [\boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in \boldsymbol{S}_1(\lambda_1) \cdot \boldsymbol{S}_2(\lambda_2)] \text{ */} \\
\quad \left\{ \langle \lambda, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \boldsymbol{c}_1 = \langle c_1, g_1\rangle \text{ and } \lambda_1' = \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \text{ and } F_1 = \{\, n \in N_1 \mid c_1(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_2 = \langle c_2, g_2\rangle \text{ and } \lambda_2' = \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \text{ and } F_2 = \{\, n \in N_2 \mid c_2(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in \boldsymbol{S}_1(\lambda_1) \cdot \boldsymbol{S}_2(\lambda_2) \text{ and } \lambda = \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \end{array} \right\} \\
= \text{/* By Definition 36 of } \odot \text{ (informally on page 93) */} \\
\quad \left\{ \langle \lambda, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \boldsymbol{c}_1 = \langle c_1, g_1\rangle \text{ and } \lambda_1' = \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \text{ and } F_1 = \{\, n \in N_1 \mid c_1(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_2 = \langle c_2, g_2\rangle \text{ and } \lambda_2' = \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \text{ and } F_2 = \{\, n \in N_2 \mid c_2(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\lambda) \text{ and } \lambda = \langle \lambda_1, \lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \end{array} \right\} \\
= \text{/* Because, by Definition 37 of } \otimes \text{ (informally on page 93), } [\lambda = \langle \lambda_1,\lambda_2\rangle \in \Lambda_1 \times \Lambda_2 \text{ and } \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\lambda) \text{ and } \lambda_1' = \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1) \text{ and } \lambda_2' = \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2)] \text{ iff } [\langle \lambda_1',\lambda_2'\rangle = (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)(\lambda, \boldsymbol{c}_1 \cup \boldsymbol{c}_2)] \text{ */} \\
\quad \left\{ \langle \lambda, F_1 \cup F_2, g_1 \wedge g_2, \langle \lambda_1',\lambda_2'\rangle\rangle \;\middle|\; \begin{array}{l} \boldsymbol{c}_1 = \langle c_1, g_1\rangle \text{ and } F_1 = \{\, n \in N_1 \mid c_1(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_2 = \langle c_2, g_2\rangle \text{ and } F_2 = \{\, n \in N_2 \mid c_2(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\lambda) \text{ and } \lambda \in \Lambda_1 \times \Lambda_2 \text{ and } \langle \lambda_1',\lambda_2'\rangle = (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)(\lambda, \boldsymbol{c}_1 \cup \boldsymbol{c}_2) \end{array} \right\} \\
= \text{/* By introducing } F = F_1 \cup F_2, \text{ and by applying } \langle \lambda_1',\lambda_2'\rangle = (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)(\lambda, \boldsymbol{c}_1 \cup \boldsymbol{c}_2) \text{ */} \\
\quad \left\{ \langle \lambda, F, g_1 \wedge g_2, (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)(\lambda, \boldsymbol{c}_1 \cup \boldsymbol{c}_2)\rangle \;\middle|\; \begin{array}{l} \boldsymbol{c}_1 = \langle c_1, g_1\rangle \text{ and } \boldsymbol{c}_2 = \langle c_2, g_2\rangle \text{ and} \\ F = \{\, n \in N_1 \cup N_2 \mid c_1(n) = \text{——} \text{ or } c_2(n) = \text{——} \,\} \text{ and} \\ \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\lambda) \text{ and } \lambda \in \Lambda_1 \times \Lambda_2 \end{array} \right\} \\
= \text{/* Because, by Definition 12 of } \cup, \; [c_1(n) = \text{——} \text{ or } c_2(n) = \text{——}] \text{ iff } [(c_1 \cup c_2)(n) = \text{——}], \text{ and by applying } \langle c, g\rangle = \boldsymbol{c} = \boldsymbol{c}_1 \cup \boldsymbol{c}_2 = \langle c_1 \cup c_2, g_1 \wedge g_2\rangle \text{ */} \\
\quad \{\, \langle \lambda, F, g, (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)(\lambda, \boldsymbol{c})\rangle \mid F = \{\, n \in N_1 \cup N_2 \mid c(n) = \text{——} \,\} \text{ and } \boldsymbol{c} = \langle c, g\rangle \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\lambda) \text{ and } \lambda \in \Lambda_1 \times \Lambda_2 \,\} \\
= \text{/* By the definition of } R' \text{ in Lemma 2 */} \\
\quad R'
\end{array}
$$

Figure 9: Proof: $R = R'$.

5 From α-Connectors to $\boldsymbol{\varepsilon}$-Connectors

In this section, we demonstrate a correspondence between $\boldsymbol{\varepsilon}$-connectors and α-connectors in the direction opposite to the previous section's: from the latter to the former. Our approach, however, resembles our approach in Section 4: we present a unary operator, denoted by $\frac{1}{\mathbb{L}}$, which takes as argument an α-connector and produces an equivalent—i.e., bi-similar—$\boldsymbol{\varepsilon}$-connector. We call our process of transforming an α-connector to an η-connector the $\frac{1}{\mathbb{L}}$-*transformation* and define the $\frac{1}{\mathbb{L}}$-operator for *any* α-connector. It follows that the class of connectors that we can model as α-connector includes those that we can model as $\boldsymbol{\varepsilon}$-connector. Since the previous section gave us a similar result in the opposite direction, we conclude that $\boldsymbol{\varepsilon}$-connectors and α-connectors have the same degree of expressiveness.

The $\frac{1}{\mathbb{L}}$ operator works as follows; suppose we wish to transform an α-connector $\mathscr{C}^{\mathsf{CA}} = \langle \sigma, \alpha \rangle$ over $[N,G]$. Whereas the connector structure σ does not incur any change (because $\frac{1}{\mathbb{L}}$ alters only the behavioral model), from the CA α, the $\frac{1}{\mathbb{L}}$-operator derives an initialized constraint next function: for each transition $\langle q,F,g,q' \rangle$ in the transition relation of α, $\frac{1}{\mathbb{L}}$ includes a mapping from state q and a constraint coloring $\boldsymbol{c} = \langle c,g \rangle$ to state q', where c assigns the flow color to all and only nodes in F.

Definition 34 (`col`)**.** *Let $N,F \subseteq \mathsf{Node}$. Then:*

$$\mathtt{col}(N,F) = \left\{ n \mapsto \kappa \;\middle|\; n \in N \text{ and } \kappa = \begin{pmatrix} \text{——} & \text{if } n \in F \\ \text{- - - -} & \text{otherwise} \end{pmatrix} \right\}$$

Definition 35 ($\frac{1}{\mathbb{L}}$ for α-connectors)**.** *Let $\mathscr{C}^{\mathsf{CA}} = \langle \sigma, \alpha \rangle$ be an α-connector over $[N,G]$ with $\alpha = \langle Q,R,q_0 \rangle$ a CA over $[N,G]$. The $\frac{1}{\mathbb{L}}$-transformation of $\mathscr{C}^{\mathsf{CA}}$, denoted by $\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$, is defined as:*

$$\tfrac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}}) = \langle \sigma, \tfrac{1}{\mathbb{L}}(\alpha) \rangle$$

$$\begin{array}{l} \textit{with: } \frac{1}{\mathbb{L}}(\alpha) = \langle \boldsymbol{\eta}, q_0 \rangle \\ \textit{and: } \boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g \rangle \rangle \mapsto q' \mid \langle q,F,g,q' \rangle \in R \,\} \end{array}$$

The following proposition states that the application of $\frac{1}{\mathbb{L}}$ to an α-connector yields an $\boldsymbol{\varepsilon}$-connector.

Proposition 2. *Let $\mathscr{C}^{\mathsf{CA}} = \langle \sigma, \alpha \rangle$ be an α-connector over $[N,G]$. Then, $\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$ is an $\boldsymbol{\varepsilon}$-connector over $[N,\boldsymbol{S}]$ with $\boldsymbol{S}$ defined over $[N,G,Q]$ as:*

$$\boldsymbol{S} = \{\, q \mapsto \boldsymbol{T} \mid q \in Q \text{ and } \boldsymbol{T} = \{\, \langle \mathtt{col}(N,F), g \rangle \mid \langle q,F,g,q' \rangle \in R \,\} \,\}$$

Proof. Let $\mathscr{C}^{\mathsf{CA}} = \langle \sigma, \alpha \rangle$ with $\alpha = \langle Q,R,q_0 \rangle$ a CA over $[N,G]$. Then, by Definition 35, $\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}}) = \langle \sigma, \frac{1}{\mathbb{L}}(\alpha) \rangle = \langle \boldsymbol{\eta}, q_0 \rangle$. By Definition 29, we must show that $\boldsymbol{\eta}$ is a constraint next function over $\boldsymbol{S}$. First, by Definition 34, all colorings that occur in the domain of $\boldsymbol{\eta}$ have N as their domain. Next, by Definition 35, all indexes that occur in the domain and co-domain of $\boldsymbol{\eta}$ are states that appear in elements of the transition relation R; therefore, by Definition 20, all indexes come from Q. Similarly, by Definition 35, all data constraints that occur in the domain of $\boldsymbol{\eta}$ also appear in elements of R, hence come from G. Finally, by Definition 27, we must show that $[\langle \lambda, \boldsymbol{c} \rangle \mapsto \lambda'] \in \boldsymbol{\eta}$ iff $\boldsymbol{c} \in \boldsymbol{S}(\lambda)$. This follows straightforwardly from Definition 35 and the definition of $\boldsymbol{S}$ in this proposition. □

5.1 Inverse

Having defined $\frac{1}{\mathbb{L}}$, we proceed by proving that it forms the *inverse* of $\mathbb{L}$ (as already hinted at by its symbol) and vice versa. We do this before stating the correctness of $\frac{1}{\mathbb{L}}$ and its distributivity over composition, because the proofs of these lemmas become significantly easier (and shorter) once we know that $\frac{1}{\mathbb{L}}$ inverts $\mathbb{L}$. The following two lemmas state the inversion properties in both directions.

Lemma 3. *Let $\mathscr{C}^{\mathsf{Col}}$ be an $\boldsymbol{\varepsilon}$-connector. Then, $\frac{1}{\mathbb{L}}(\mathbb{L}(\mathscr{C}^{\mathsf{Col}})) = \mathscr{C}^{\mathsf{Col}}$.*

Proof. Let $\mathscr{C}^{\mathsf{Col}} = \langle \sigma, \boldsymbol{\varepsilon} \rangle$ be defined over $[N,\boldsymbol{S}]$ with $\boldsymbol{S}$ a constraint CTM over $[N,G,\Lambda]$. By Definitions 29, 32, and 35, we must show that $\frac{1}{\mathbb{L}}(\mathbb{L}(\boldsymbol{\varepsilon})) = \boldsymbol{\varepsilon}$. This follows from Figure 10. □

Lemma 4. *Let $\mathscr{C}^{\mathsf{CA}}$ be an α-connector. Then, $\mathbb{L}(\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})) = \mathscr{C}^{\mathsf{CA}}$.*

Proof. Let $\mathscr{C}^{\mathsf{CA}} = \langle \sigma, \alpha \rangle$ be defined over $[N,G]$. By Definitions 21, 32, and 35, we must show that $\mathbb{L}(\frac{1}{\mathbb{L}}(\alpha)) = \alpha$. This follows from Figure 11. □

$\frac{1}{\mathbb{L}}(\mathbb{L}(\boldsymbol{\varepsilon}))$
$=$ /* By Definition 32 of $\mathbb{L}$, and because $\boldsymbol{\varepsilon}$ is defined over $\boldsymbol{S}$, which is defined over $[N,G,\Lambda]$ */
$\frac{1}{\mathbb{L}}(\langle \Lambda, R, \lambda_0 \rangle)$ with R as in Definition 32
$=$ /* By Definition 35 of $\frac{1}{\mathbb{L}}$, and because $\mathbb{L}(\boldsymbol{\varepsilon})$ is an α-connector over $[N,G]$ by Proposition 1 */
$\langle \{\, \langle \lambda, \langle \mathtt{col}(N,F), g \rangle \rangle \mapsto \lambda' \mid \langle \lambda, F, g, \lambda' \rangle \in R \,\}, \lambda_0 \rangle$ with R as in Definition 32
$=$ /* Because $\langle \lambda, F, g, \lambda' \rangle \in R$ iff $[\lambda \in \Lambda$ and $\boldsymbol{c} = \langle c, g \rangle \in \boldsymbol{S}(q)$ and
$F = \{\, n \in N \mid c(n) = \text{——} \,\}$ and $\lambda' = \boldsymbol{\eta}(\lambda, \boldsymbol{c})]$ by Definition 32 of $\mathbb{L}$ */
$\langle \{\, \langle \lambda, \langle \mathtt{col}(N,F), g \rangle \rangle \mapsto \boldsymbol{\eta}(\lambda, \boldsymbol{c}) \mid \lambda \in \Lambda$ and $\boldsymbol{c} = \langle c, g \rangle \in \boldsymbol{S}(\lambda)$ and $F = \{\, n \in N \mid c(n) = \text{——} \,\} \,\}, \lambda_0 \rangle$
$=$ /* Because $c = \mathtt{col}(N,F)$ iff $F = \{\, n \in N \mid c(n) = \text{——} \,\}$ by Definitions 5 and 34 */
$\langle \{\, \langle \lambda, \langle c, g \rangle \rangle \mapsto \boldsymbol{\eta}(\lambda, \boldsymbol{c}) \mid \lambda \in \Lambda$ and $\boldsymbol{c} = \langle c, g \rangle \in \boldsymbol{S}(\lambda) \,\}, \lambda_0 \rangle$
$=$ /* By Definitions 27 and 28 */
$\langle \boldsymbol{\eta}, \lambda_0 \rangle = \boldsymbol{\varepsilon}$

Figure 10: Proof: $\frac{1}{\mathbb{L}}(\mathbb{L}(\boldsymbol{\varepsilon})) = \boldsymbol{\varepsilon}$.

5.2 Correctness and Distributivity of $\frac{1}{\mathbb{L}}$

As mentioned previously, knowing that $\mathbb{L}(\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})) = \mathscr{C}^{\mathsf{CA}}$ (with $\mathscr{C}^{\mathsf{CA}}$ an α-connector) greatly simplifies our correctness and distributivity proofs. We start with the former. Lemma 5, which appears below, states the bi-similarity between $\mathscr{C}^{\mathsf{CA}}$ and its $\frac{1}{\mathbb{L}}$-transformation $\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$. In addition to the inversion lemma, in our proof, we apply the bi-similarity lemma of $\mathbb{L}$.

Lemma 5. *Let $\mathscr{C}^{\mathsf{CA}}$ be an α-connector. Then, $\mathscr{C}^{\mathsf{CA}} \sim \frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$.*

Proof. Let $\mathscr{C}^{\mathsf{Col}} = \frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$. Then, $\mathscr{C}^{\mathsf{CA}} \sim \frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$ iff $\mathbb{L}(\frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})) \sim \frac{1}{\mathbb{L}}(\mathscr{C}^{\mathsf{CA}})$ iff $\mathbb{L}(\mathscr{C}^{\mathsf{Col}}) \sim \mathscr{C}^{\mathsf{Col}}$ by Lemma 4. The latter, $\mathbb{L}(\mathscr{C}^{\mathsf{Col}}) \sim \mathscr{C}^{\mathsf{Col}}$, follows from Lemma 1. □

Finally, Lemma 6 states the distributivity of $\frac{1}{\mathbb{L}}$ over composition: informally, this means that it does not matter whether we first compose α-connectors $\mathscr{C}_1^{\mathsf{CA}}$ and $\mathscr{C}_2^{\mathsf{CA}}$ and then apply $\frac{1}{\mathbb{L}}$ to the composition or first apply $\frac{1}{\mathbb{L}}$ to $\mathscr{C}_1^{\mathsf{CA}}$ and $\mathscr{C}_2^{\mathsf{CA}}$ and then compose the transformations; the resulting $\boldsymbol{\varepsilon}$-connectors equal each other. Our proof, similar to that of the previous lemma, relies on the inversion lemmas.

Lemma 6. *Let $\mathscr{C}_1^{\mathsf{CA}}$ and $\mathscr{C}_2^{\mathsf{CA}}$ be α-connectors. Then, $\frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}}) \bowtie \frac{1}{\mathbb{L}}(\mathscr{C}_2^{\mathsf{CA}}) = \frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}} \times \mathscr{C}_2^{\mathsf{CA}})$.*

Proof. Follows from Figure 12. □

6 Application

In this section, we sketch an application of the results presented above: the integration of verification and animation of context-sensitive connectors in Vereofy [5], a model checking tool for α-connectors that operates on constraint automata.[9] Broadly, this application consists of two parts: model checking connectors built from context-sensitive constituents and generating animated counterexamples.

Verification of $\boldsymbol{\varepsilon}$-connectors Vereofy operates on constraint automata and, therefore, many consider it unable to verify context-sensitive connectors. We mend this deficiency as follows. First, we note that recent research established that one can transform coloring models with three colors, known for their ability to properly capture context-sensitivity, to corresponding coloring models with two colors [10]. Essentially, this means that $\boldsymbol{\varepsilon}$-connectors as defined in this paper—i.e., featuring only two colors—can

[9] Vereofy is freely available on-line at: `http://www.vereofy.de`.

$\mathbb{L}(\frac{1}{\mathbb{L}}(\alpha))$

$=$ /* By Definition 35 of $\frac{1}{\mathbb{L}}$, and because $\alpha = \langle Q,R,q_0\rangle$ is a constraint automaton over $[N,G]$ by the premise of Lemma 4 */
$\mathbb{L}(\langle \boldsymbol{\eta}, q_0\rangle)$ with $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$

$=$ /* By Definition 32 of $\mathbb{L}$, and because $\boldsymbol{\eta}$ is defined over $\boldsymbol{S}$ with $\boldsymbol{S} = \{\, q \mapsto \boldsymbol{T} \mid q \in Q$ and $\boldsymbol{T} = \{\, \langle \mathtt{col}(N,F), g\rangle \mid \langle q,F,g,q'\rangle \in R \,\}\,\}$ by Proposition 2 */
$\langle Q,R',q_0\rangle$ with:
- $R' = \{\, \langle q,F,g,\boldsymbol{\eta}(q,\boldsymbol{c})\rangle \mid q \in Q$ and $\boldsymbol{c} = \langle c,g\rangle \in \boldsymbol{S}(q)$ and $F = \{\, n \in N \mid c(n) = \text{——} \,\}\,\}$
- $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$
- $\boldsymbol{S} = \{\, q \mapsto \boldsymbol{T} \mid q \in Q$ and $\boldsymbol{T} = \{\, \langle \mathtt{col}(N,F), g\rangle \mid \langle q,F,g,q'\rangle \in R \,\}\,\}$

$=$ /* By introducing $q' = \boldsymbol{\eta}(q,\boldsymbol{c})$ in R' */
$\langle Q,R',q_0\rangle$ with:
- $R' = \{\, \langle q,F,g,q'\rangle \mid q \in Q$ and $\boldsymbol{c} = \langle c,g\rangle \in \boldsymbol{S}(q)$ and $F = \{\, n \in N \mid c(n) = \text{——} \,\}$ and $q' = \boldsymbol{\eta}(q,\boldsymbol{c}) \,\}$
- $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$
- $\boldsymbol{S} = \{\, q \mapsto \boldsymbol{T} \mid q \in Q$ and $\boldsymbol{T} = \{\, \langle \mathtt{col}(N,F), g\rangle \mid \langle q,F,g,q'\rangle \in R \,\}\,\}$

$=$ /* Because, by the definition of $\boldsymbol{S}$, $[\boldsymbol{c} = \langle c,g\rangle \in \boldsymbol{S}(q)]$ iff $[\langle q,F',g,q''\rangle \in R$ and $c = \mathtt{col}(N,F')]$ */
$\langle Q,R',q_0\rangle$ with:
- $R' = \{\, \langle q,F,g,q'\rangle \mid q \in Q$ and $\langle q,F',g,q''\rangle \in R$ and $c = \mathtt{col}(N,F')$ and $F = \{\, n \in N \mid c(n) = \text{——} \,\}$ and $q' = \boldsymbol{\eta}(q,\langle c,g\rangle) \,\}$
- $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$

/* By applying $c = \mathtt{col}(N,F')$ */
- $R' = \{\, \langle q,F,g,q'\rangle \mid q \in Q$ and $\langle q,F',g,q''\rangle \in R$ and $F = \{\, n \in N \mid (\mathtt{col}(N,F'))(n) = \text{——} \,\}$ and $q' = \boldsymbol{\eta}(q,\langle \mathtt{col}(N,F'), g\rangle) \,\}$
- $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$

/* Because, by Definition 34 of col, $\{\, n \in N \mid (\mathtt{col}(N,F'))(n) = \text{——} \,\} = F'$ */
$\langle Q,R',q_0\rangle$ with:
- $R' = \{\, \langle q,F,g,q'\rangle \mid q \in Q$ and $\langle q,F',g,q''\rangle \in R$ and $F = F'$ and $q' = \boldsymbol{\eta}(q,\langle \mathtt{col}(N,F'), g\rangle) \,\}$
- $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$

/* By applying $F = F'$ */
$\langle Q,R',q_0\rangle$ with:
- $R' = \{\, \langle q,F',g,q'\rangle \mid q \in Q$ and $\langle q,F',g,q''\rangle \in R$ and $q' = \boldsymbol{\eta}(q,\langle \mathtt{col}(N,F'), g\rangle) \,\}$
- $\boldsymbol{\eta} = \{\, \langle q, \langle \mathtt{col}(N,F), g\rangle\rangle \mapsto q' \mid \langle q,F,g,q'\rangle \in R \,\}$

$=$ /* Because, by the definition of $\boldsymbol{\eta}$, $q' = \boldsymbol{\eta}(q,\langle \mathtt{col}(N,F'), g\rangle)$ iff $\langle q,F',g,q'\rangle \in R$ */
$\langle Q,R',q_0\rangle$ with $R' = \{\, \langle q,F',g,q'\rangle \mid q \in Q$ and $\langle q,F',g,q''\rangle \in R$ and $\langle q,F',g,q'\rangle \in R \,\} = R$

$=$ /* By the definition of α */
α

Figure 11: Proof: $\mathbb{L}(\frac{1}{\mathbb{L}}(\alpha)) = \alpha$.

serve as faithful models of context-sensitive circuits. Consequently, the results in Section 4 enable the verification of such connectors with Vereofy: using the $\mathbb{L}$-transformation, we transform context-sensitive $\boldsymbol{\varepsilon}$-connectors to context-sensitive α-connectors, whose CA we subsequently can analyze with Vereofy. In this application, the distributivity of $\mathbb{L}$ over composition in Lemma 2 plays an important role (as already outlined in Section 4.2): it facilitates (i) the one-time-application of $\mathbb{L}$ to the context-sensitive $\boldsymbol{\varepsilon}$-connectors of Reo's primitives after which (ii) we can use Vereofy's built-in functionality for CA composition to construct the complex automata that we wish to inspect. Examples appear in [10]. The distributivity lemmas work also in the opposite direction: if future studies indicate that composition of coloring models costs less than composition of CA, we may extend Vereofy with a module to automatically (1) transform CA of primitives to coloring models with $\frac{1}{\mathbb{L}}$, (2) compose the resulting coloring models, and (3) transform the resulting composition back to a CA with $\mathbb{L}$. (To truly gain in performance, however, the costs of transforming forth and back should not exceed the benefits of composing coloring models instead of CA.)

$\frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}}) \times \frac{1}{\mathbb{L}}(\mathscr{C}_2^{\mathsf{CA}})$
$= /^*$ By the inversion of $\mathbb{L}$ by $\frac{1}{\mathbb{L}}$ in Lemma 3 $*/$
$\frac{1}{\mathbb{L}}(\mathbb{L}(\frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}}) \times \frac{1}{\mathbb{L}}(\mathscr{C}_2^{\mathsf{CA}})))$
$= /^*$ By the distributivity of $\mathbb{L}$ over composition in Lemma 2 $*/$
$\frac{1}{\mathbb{L}}(\mathbb{L}(\frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}})) \times \mathbb{L}(\frac{1}{\mathbb{L}}(\mathscr{C}_2^{\mathsf{CA}})))$
$= /^*$ By the inversion of $\frac{1}{\mathbb{L}}$ by $\mathbb{L}$ in Lemma 4 $*/$
$\frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}} \times \mathscr{C}_2^{\mathsf{CA}})$

Figure 12: Proof: $\frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}}) \times \frac{1}{\mathbb{L}}(\mathscr{C}_2^{\mathsf{CA}}) = \frac{1}{\mathbb{L}}(\mathscr{C}_1^{\mathsf{CA}} \times \mathscr{C}_2^{\mathsf{CA}})$.

Animation of α-connectors Vereofy facilitates the generation and inspection of counterexamples, an important feature that distinguishes it from mCRL2 (another tool sometimes used for model checking Reo circuits [12]).[10] When using Vereofy in conjunction with the Eclipse Coordination Tools (Reo's standard distribution),[11] it can in *some* cases display counterexamples as connector animations. These animated counterexamples comprise a graphical model of a connector (similar to Figure 1) through which data items visually flow for each computation step of a faulty behavior. Although this approach greatly enhances the ease with which users can analyze counterexamples, the opportunity to actually provide these animations depends on the availability of a coloring model of the connector under investigation (in addition to the constraint automaton that Vereofy's verification algorithm operates on). Moreover, the standalone version of Vereofy, a command-line tool, does not facilitate the animation of counterexamples at all. The results in Section 5, however, enable animated counterexamples for *any* CA: in the case of unavailability of a coloring model, Vereofy can simply generate such a model with the $\frac{1}{\mathbb{L}}$-transformation.

7 Concluding Remarks

Related work Closest to the work in this paper seems an informal discussion in [8, 9] on the equivalence of coloring models and constraint automata. These cited references, however, do not support their claims with formal evidence, nor do they provide an algorithm, operation, or function to actually transform connector models back and forth. More generally, we know of only a few other correspondences between different semantic models of Reo connectors, the oldest concerning CA and coalgebraic models: the set of runs of an α-connector $\mathscr{C}^{\mathsf{CA}}$ corresponds to the set of *timed data streams* induced by the coalgebraic model of the same circuit that $\mathscr{C}^{\mathsf{CA}}$ models (Definition 3.6 in [6]). Reo's semantics in terms of the *unifying theories of programming* [13] appears closely related to the coalgebraic semantics as well, but we do not know of any formal claims or proofs. Two other correspondences concern *tile models* [3] and coloring models: Arbab et al. prove in Theorems 1 and 3 of [3] that a semantic model based on *tiles*, which resemble colorings, yields behavioral formalisms equal to coloring models with two or three colors (depending on the tile definitions).

Conclusion and future work We showed that, once extended with data constraints, coloring models with two colors and constraint automata have the same degree of expressiveness by defining two operators that transform data-aware coloring models to equivalent CA and vice versa. Moreover, these operators distribute over composition, a desirable property especially from a practical point of view. Though primarily a theoretical contribution, we illustrated how our results can broaden the applicability

[10] mCRL2 is freely available on-line at: `http://www.mcrl2.org`.
[11] The Eclipse Coordination Tools are freely available on-line at `http://reo.project.cwi.nl`.

of Reo's tools. With respect to future work, we would like to implement the transformation operators and the sketched extension to Vereofy. Another application worth investigation comprises the development of an implementation of Reo based on transforming the behavioral models of connectors back and forth. Finally, we would like to study correspondences between other semantic models (e.g., guarded automata [7] and intentional automata [9]).

Acknowledgments We would like to thank the reviewers and the members of the ICE 2011 discussion forum `gege`, `wind`, `wolf` and `xyz` for their valuable comments.

References

[1] Farhad Arbab (2004): *Reo: A channel-based coordination model for component composition*. *Mathematical Structures in Computer Science* 14(3), pp. 329–366, doi:10.1017/S0960129504004153.

[2] Farhad Arbab (2005): *Abstract behavior types: A foundation model for components and their composition*. *Science of Computer Programming* 55(1–3), pp. 3–52, doi:10.1016/j.scico.2004.05.010.

[3] Farhad Arbab, Roberto Bruni, Dave Clarke, Ivan Lanese & Ugo Montanari (2009): *Tiles for Reo*. In Andrea Corradini & Ugo Montanari, eds.: *Recent Trends in Algebraic Development Techniques, LNCS* 5486, Springer, pp. 37–55, doi:10.1007/978-3-642-03429-9_4.

[4] Farhad Arbab & Jan Rutten (2003): *A coinductive calculus of component connectors*. In Marin Wirsing, Dirk Pattinson & Rolf Hennicker, eds.: *Recent Trends in Algebraic Development Techniques, LNCS* 2755, Springer, pp. 34–55, doi:10.1007/978-3-540-40020-2_2.

[5] Christel Baier, Tobias Blechmann, Joachim Klein & Sascha Klüppelholz (2009): *Formal verification for components and connectors*. In Frank de Boer, Marcello Bonsangue & Eric Madelaine, eds.: *Formal Methods for Components and Objects, LNCS* 5751, Springer, pp. 82–101, doi:10.1007/978-3-642-04167-9_5.

[6] Christel Baier, Marjan Sirjani, Farhad Arbab & Jan Rutten (2006): *Modeling component connectors in Reo by constraint automata*. *Science of Computer Programming* 61(2), pp. 75–113, doi:10.1016/j.scico.2005.10.008.

[7] Marcello Bonsangue, Dave Clarke & Alexandra Silva (2009): *Automata for context-dependent connectors*. In John Field & Vasco Vasconcelos, eds.: *Coordination Models and Languages, LNCS* 5521, Springer, pp. 184–203, doi:10.1007/978-3-642-02053-7_10.

[8] Dave Clarke, David Costa & Farhad Arbab (2007): *Connector colouring I: Synchronisation and context dependency*. *Science of Computer Programming* 66(3), pp. 205–225, doi:10.1016/j.scico.2007.01.009.

[9] David Costa (2010): *Formal Models for Component Connectors*. Ph.D. thesis, Vrije Universiteit Amsterdam.

[10] Sung-Shik Jongmans, Christian Krause & Farhad Arbab (2011): *Encoding context-sensitivity in Reo into non-context-sensitive semantic models*. In Wolfgang de Meuter & Catalin Roman, eds.: *Proceedings of the 13th International Conference on Coordination Models and Languages, LNCS* 6721, Springer, pp. 31–48, doi:10.1007/978-3-642-21464-6_3.

[11] Christian Koehler & Dave Clarke (2009): *Decomposing port automata*. In: *Proceedings of the 2009 ACM Symposium on Applied Computing*, pp. 1369–1373, doi:10.1145/1529282.1529587.

[12] Natallia Kokash, Christian Krause & Erik de Vink (2010): *Verification of context-dependent channel-based service models*. In Frank de Boer, Marcello Bonsangue, Stefan Hallerstede & Michael Leuschel, eds.: *Formal Methods for Components and Objects, LNCS* 6286, Springer, pp. 21–40, doi:10.1007/978-3-642-17071-3_2.

[13] Sun Meng & Farhad Arbab (2009): *Connectors as designs*. *ENTCS* 255, pp. 119–135, doi:10.1016/j.entcs.2009.10.028.

[14] José Proença (2011): *Synchronous Coordination of Distributed Components*. Ph.D. thesis, Universiteit Leiden.

A Appendix: Composition Operators

In this appendix, we give the formal definitions of the composition operators whose definition we gave only informally in Section 3. More specifically, we give the definitions of the composition operators for constraint CTMs, (initialized) constraint next functions, and $\boldsymbol{\varepsilon}$-connectors. The definitions of the composition operators for constraint colorings and constraint coloring tables appear in Section 3. As mentioned in that section, we obtain the operators that we define below by replacing S_1, S_2, η_1, η_2, ε_1, and ε_2 in Definitions 14–17 by their $\boldsymbol{font}$ versions $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_2$, $\boldsymbol{\varepsilon}_1$, and $\boldsymbol{\varepsilon}_2$.

Definition 36 (Composition of constraint CTMs)**.** *Let $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ be constraint CTMs over $[N_1, G_1, \Lambda_1]$ and $[N_2, G_2, \Lambda_2]$. Their composition, denoted by $\boldsymbol{S}_1 \odot \boldsymbol{S}_2$, is a constraint CTM over $[N_1 \cup N_2, G_1 \wedge G_2, \Lambda_1 \times \Lambda_2]$[5] defined as:*

$$\boldsymbol{S}_1 \odot \boldsymbol{S}_2 = \{\, \langle \lambda_1, \lambda_2 \rangle \mapsto \boldsymbol{S}_1(\lambda_1) \cdot \boldsymbol{S}_2(\lambda_2) \mid \lambda_1 \in \Lambda_1 \text{ and } \lambda_2 \in \Lambda_2 \,\}$$

Definition 37 (Composition of constraint next functions)**.** *Let $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ be constraint next functions over $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ with $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ constraint CTMs over $[N_1, G_1, \Lambda_1]$ and $[N_2, G_2, \Lambda_2]$. Their composition, denoted by $\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2$, is a next function over $\boldsymbol{S}_1 \odot \boldsymbol{S}_2$ defined as:*

$$\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2 = \left\{ \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle, \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \\ \downarrow \\ \langle \boldsymbol{\eta}_1(\lambda_1, \boldsymbol{c}_1), \boldsymbol{\eta}_2(\lambda_2, \boldsymbol{c}_2) \rangle \end{array} \middle| \begin{array}{c} \langle \lambda_1, \lambda_2 \rangle \in \Lambda_1 \times \Lambda_2 \\ \textit{and} \\ \boldsymbol{c}_1 \cup \boldsymbol{c}_2 \in (\boldsymbol{S}_1 \odot \boldsymbol{S}_2)(\langle \lambda_1, \lambda_2 \rangle) \end{array} \right\}$$

Definition 38 (Composition of initialized constraint next functions)**.** *Let $\boldsymbol{\varepsilon}_1 = \langle \boldsymbol{\eta}_1, \lambda_0^1 \rangle$ and $\boldsymbol{\varepsilon}_2 = \langle \boldsymbol{\eta}_2, \lambda_0^2 \rangle$ be initialized constraint next functions over $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. Their composition, denoted by $\boldsymbol{\varepsilon}_1 \otimes \boldsymbol{\varepsilon}_2$, is an initialized next function over $[\boldsymbol{S}_1 \odot \boldsymbol{S}_2]$ defined as:*

$$\boldsymbol{\varepsilon}_1 \otimes \boldsymbol{\varepsilon}_2 = \langle \boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2, \langle \lambda_0^1, \lambda_0^2 \rangle \rangle$$

Definition 39 (Composition of $\boldsymbol{\varepsilon}$-connectors)**.** *Let $\mathscr{C}_1^{\mathsf{Col}} = \langle \sigma_1, \boldsymbol{\varepsilon}_1 \rangle$ and $\mathscr{C}_2^{\mathsf{Col}} = \langle \sigma_2, \boldsymbol{\varepsilon}_2 \rangle$ be $\boldsymbol{\varepsilon}$-connectors over $[N_1, \boldsymbol{S}_1]$ and $[N_2, \boldsymbol{S}_2]$ such that $\sigma_1 \boxtimes \sigma_2$ is defined. Their composition, denoted by $\mathscr{C}_1^{\mathsf{Col}} \times \mathscr{C}_2^{\mathsf{Col}}$, is an $\boldsymbol{\varepsilon}$-connector over $[N_1 \cup N_2, \boldsymbol{S}_1 \odot \boldsymbol{S}_2]$ defined as:*

$$\mathscr{C}_1^{\mathsf{Col}} \times \mathscr{C}_2^{\mathsf{Col}} = \langle \sigma_1 \boxtimes \sigma_2, \boldsymbol{\varepsilon}_1 \otimes \boldsymbol{\varepsilon}_2 \rangle$$